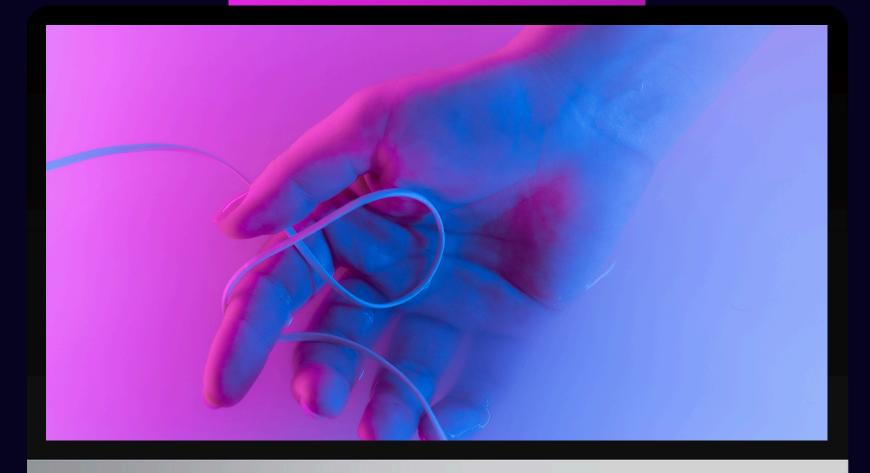


KKESERCITAZIONEESERCIZIO_PRATICO_U3_W3_ L2



TRACCIA

Lo scopo dell'esercizio di oggi è di acquisire esperienza con IDA, un tool fondamentale per l'a n a lis i s t a t ic a . A tal proposito, con riferimento al malware chiamato «Malware_U3_W3_L2 » presente all'interno della cartella «Esercizio_Pratico_U3_W3_L2 » sul Desktop della macchina virtuale dedicata all'analisi dei malware, rispondere ai seguenti quesiti, utilizzando IDA Pro.

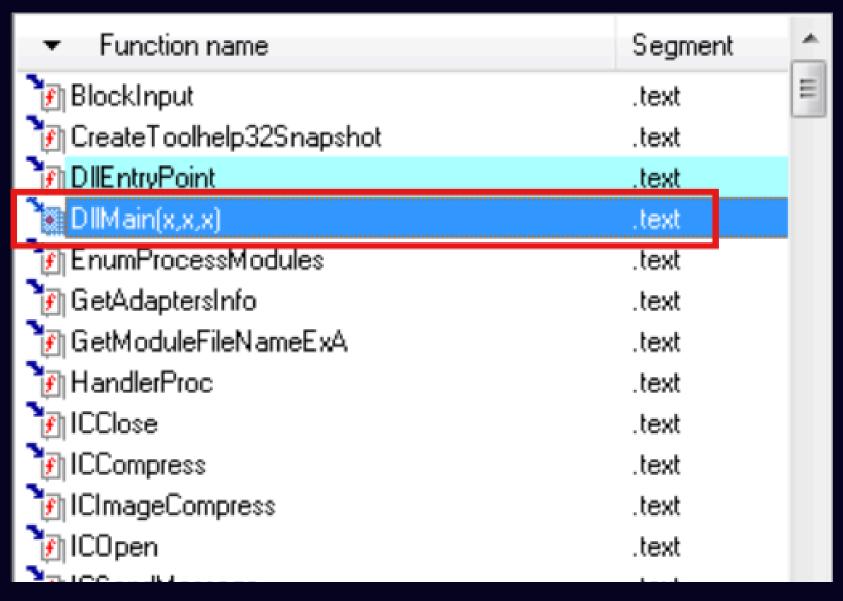
- 1. Individuare l'indirizzo della funzione DLLMain (così com'è, in esadecimale)
- 2. Dalla scheda «imports» individuare la funzione «gethostbyname ». Qual è l'indirizzo dell'import? Cosa fa la funzione?
- 3. Quante sono le variabili locali della funzione alla locazione di memoria 0x10001656?
- 4. Quanti sono, invece, i parametri della funzione sopra? 5. Inserire altre considerazioni macro livello sul malware (comportamento)

Home



INDIVIDUARE L'INDIRIZZO DELLA FUNZIONE DLLMAIN (COSÌ COM'È, IN ESADECIMALE)

1000D02E	8B	44	24	08	48	ØF	85	CE	00	99	00	8B	44	24	64	53	ΪD\$.H.à+ΫC
1000D03E	A3	99	30	99	10	A1	44	90	01	10	56	83	CO	ØD	57	50	ú.0íDÉVâ+
1000D04E	E8	F9	7E	99	99	8B	1D	98	62	91	10	8B	35	CO	62	91	Þ"~ïbï5
1000D05E	10	33	FF	59	85	CO	74	23	A1	44	90	91	10	6A	07	83	.3 Yà+t#íÞÉ
1000D06E	CO	ØD	68	58	39	69	10	50	FF	D6	83	C4	OC	85	CO	75	+.hX9P [â
1000D07E	ØA	57	57	57	68	74	10	99	10	ΕB	34	A1	44	90	01	10	.WWWhtÙ4íC
1000D08E	83	CØ	ØD	50	E8	B5	7E	99	99	85	CO	59	74	2A	A1	44	â+.PÞÁ~à+Yt
1000D09E	90	01	10	6A	96	83	CO	ØD	68	40	39	69	10	50	FF	D6	Éj.â+.hL9
1000D 0AE	83	C4	9C	85	CO	75	11	57	57	57	68	65	13	99	10	57	aà+u.WWWhe.
1000D 0BE	57	FF	D3	A3	04	30	69	10	A1	3C	90	01	10	8B	35	B8	W Ëú.Oí<É
1000D0CE	62	91	10	83	CO	ØD	6A	10	50	68	D8	E5	08	10	FF	D6	bâ+.j.PhÏÕ.
1000D 0DE	A1	38	90	01	10	6A	05	83	CO	ØD	50	68	E8	E5	08	10	18Éj.â+.Pht
1000D 0EE	FF	D6	83	C4	18	57	57	57	68	56	16	99	10	57	57	FF	_íâ₩WWhU
1000D0FE	D3	5F	5E	A3	08	30	69	10	5B	6A	01	58	C2	0C	00	55	Ë_^ú.0[j.X-
1000D10E	8B	EC	81	EC	99	03	99	00	53	56	57	33	F6	FF	75	98	iýüýSVW3÷
1000D11E	56	68	2A	04	99	00	FF	15	28	62	01	10	BF	99	01	99	Vh*(b
1000D12E	99	89	45	08	57	8D	85	00	FF	FF	FF	56	50	E8	12	7E	.ëE.Wìà. UF
1000D13E	99	99	FF	75	0C	8D	85	00	FF	FF	FF	50	E8	51	7E	99	u.ìà. Pt
1000D14E	99	8D	85	99	FF	FF	FF	50	E8	F1	7D	99	99	83	C4	18	.ìà. PÞ±}
1000D15E	8D	5C	99	02	8D	85	99	FD	FF	FF	57	50	8D	85	99	FF	ì\ìà.² WPì
1000D16E	FF	FF	6A	FF	50	56	56	FF	15	E4	61	91	10	6A	04	68	j PVV "õa".
1000D17E	99	10	99	99	53	56	FF	75	08	FF	15	ΕØ	61	91	10	8B	SV uÓa
1000D18E	F8	56	8D	85	99	FD	FF	FF	53	50	57	FF	75	98	FF	15	°Vìà.² SPW u
1000D19E	DC	61	01	10	68	60	4D	69	10	68	54	4D	69	10	FF	15	_ah`MhTM.
1000D1AE	F8	60	01	10	50	FF	15	99	62	01	10	56	56	57	50	56	[™] `PbVU
0000C42E	00000	0001	1000	002E	DIIN	/lain	(x,x,x)									



Per prima cosa per sapere quale sia l'indirizzo esadecimale della funzione dobbiamo andare a ricercare il nome della nostra funzione dall'elenco presente nella parte sinistra del programma IDA, come da immagine sopra.

Come potete vedere dall'immagine a sinistra, l'indirizzo esadecimale della funzione è **00000001000D02E**

Home



DALLA SCHEDA «IMPORTS» INDIVIDUARE LA FUNZIONE «GETHOSTBYNAME». QUAL È L'INDIRIZZO DELL'IMPORT? COSA FA LA FUNZIONE?

x 🖹 IDA View-A 🛮 x	Hex View-A	X ⅓ Structures
▼ Address	Ordinal	Name
₩ 00000000100163A8		wavelnOpen
₩ 00000000100163AC		waveInClose
tt 000000000100163B0		waveInUnprepareHe
tt 000000000100163B4		waveInPrepareHead
tt 000000000100163B8		wavelnAddBuffer
tt 000000000100163BC		waveInStart
tt 000000000100163C4	18	select
№ 00000000100163C8	11	inet_addr
00000000100163CC	52	gethostbyname
□ 000000000000000000000000000000000000	12	inet_ntoa

In questo caso per scoprire l'indirizzo di memoria dell'import chiamato <<**gethostbyname**>> dobbiamo recarci nella sezione **IMPORTS** e cercare la funzione desiderata come mostrato in figura, nel nostro caso l'indirizzo è **0000000100163CC**



QUANTE SONO LE VARIABILI LOCALI DELLA FUNZIONE ALLA LOCAZIONE DI MEMORIA OX10001656? QUANTI SONO, INVECE, I PARAMETRI DELLA FUNZIONE SOPRA?

var_675= byte ptr -675h var 674= dword ptr -674h hLibModule= dword ptr -6701 timeout= timeval ptr -66Ch name= sockaddr ptr -664h var 654= word ptr -654h Dst= dword ptr -650h Parameter= byte ptr -644h var 640= byte ptr -640h CommandLine= byte ptr -63Fh Source= byte ptr -63Dh Data= byte ptr -638h 637= bute ptr -637h 544= dword ptr -544h 50C= dword ptr -50Ch **500= dword ptr -**500h Buf2= bute ptr -4FCh readfds= fd set ptr -4BCh phkResult= byte ptr -3B8h var 3B0= dword ptr -3B0h 1A4= dword ptr -1A4h var 194= dword ptr -194h WSAData= WSAData ptr -190h arq 0= dword ptr

Per rispondere alle domande poste nella traccia, dobbiamo andare ad aprire la funzione **SUB_10001656**, quindi ci posizioniamo sulla scheda IDA View-A e ritorniamo sulla lista di funzioni che abbiamo visto prima e cerchiamo la funzione SUB_10001656, quindi doppio click su di essa.

Home

Il risultato del doppioclick lo trovate a sinistra, ovvero le informazioni riguardanti la nostra funzione:

Nel riquadro rosso abbiamo le variabili dato che come sappiamo in assembly i valori negativi sono le variabili locali mentre i valori positivi presenti nel riquadro arancione sono parametri

CONSIDERAZIONI COMPORTAMENTO MALWARE

Andando a verificare gli import presenti nelle funzioni sembra che il malware vada ad agire sulle chiavi di registro, sui processi del sistema, sulla memoria e sul network del sistema compromettendo le corrette funzionalità di rete



